Request for Information:
Open Source Software Development Acceleration (OSSODA)

National Nuclear Security Administration (NNSA)
Advanced Simulation and Computing (ASCI)
ASCI PathForward Program

13 May 2003

UCRL-MI-153295

Table of Contents

Introduction

The Open Source development methodology has yielded significant enhancements to the state-of-the-art in operating systems (c.f., Linux OS) and tools (c.f., Apache). Multiple accretions of open software mass have resulted in profitable enterprises that combine these tools into single offerings with support. In addition, there exist many efforts to build clustering tools (c.f., LLNL Chaos, LANL Clustermatic, Sandia CPLANT, NPACI Rocks, OSCAR and others) that extend the desktop environment to medium and high-performance computing.

The NNSA Tri-Laboratory community (LLNL, Sandia, LANL, and DOE Office of Science) has been participating in Open Source development efforts. Prior to advent of Open Source community development methodologies, the Tri-Laboratory community made significant contributions to computer science via public domain release of operating systems, code development tool chains and applications.

The NNSA Tri-Laboratory community recognizes that open source software is not "free." The cost structure is just different. This concept is described in more detail in section 1.2. In addition, NNSA recognizes that to be a full member of the open source development communities, financial and effort contributions are required.

The purpose of this Tri-Laboratory Request For Information (RFI) is to gauge the interest of Industry and the Open Source development communities in pursuing open source development efforts aimed at the needs of High Performance Technical Computing community in general and NNSA ASCI program in particular. We intend to stimulate development in specific areas (see section 2 and its associated appendices). In addition, we request information from the Industry and Open Source development communities on efforts outside of our predefined areas that may be of mutual interest.

1.1 Background

The US commitment to ending underground nuclear testing, constraints on non-nuclear testing, and loss of production capability call for new means of verifying the safety, reliability, and performance of the US nuclear stockpile. One of these means is compute-based modeling, simulation, and virtual prototyping of nuclear weapon systems. The Advanced Simulation and Computing program (aka ASCI) is one element of NNSA's Stockpile Stewardship Program (SSP) and is designed to advance NNSA's computational capabilities to help meet the future needs of stockpile stewardship. ASCI is creating leading-edge computational modeling capabilities that are essential for maintaining the safety, reliability, and performance of the stockpile. ASCI applications require near-term performance in the 10-to-30 teraFLOP/s range. Future platforms include ASCI Red Storm at 40 teraFLOP/s level in 4QFY04 (www.sandia.gov/media/NewsRel/NR2002/redstorm.htm) and with production a platform at 100 teraFLOP/s in 1QFY05 and computational science research platform at 360 teraFLOP/s with BlueGene/L (www.llnl.gov/asci/purple/, www.research.ibm.com/bluegene/).

Current trends in developing and implementing ultra-scale computers fall well below ASCI and NNSA Stockpile Stewardship Program mission requirements. ASCI

applications require a threshold shift of 100 to 1,000 times increase in computing capability. As a result, NNSA and its National Labs are pursuing this PathForward program to narrow the gap. Based on a number of factors, we believe that ultra-scale platforms will be developed using - among other things - commodity-based hardware components and open source software. It is also anticipated that open source software will impact far beyond commodity-based ultra-scale platforms. This PathForward effort is aimed at developing open source software components, technologies, and enhancements to attain the 100X to 1,000X threshold shift in computational capability and capacity that is vitally needed to meet ASCI and SSMP mission requirements. The overall ASCI PathForward program element's goal is to incentivize the US computer industry to produce key technologies, which would otherwise not be available or available later than required by ASCI that will become an integral component of potential ASCI platform RFP responses in three to five year timeframe.

In a balanced computational environment, hardware and software improvements have contributed equally to the increase in applications performance. What matters most is the shortest time to solution. Code development efforts are definitely on the critical path to success. ASCI is focused on achieving ultra-scale performance levels, suitable for solving large-scale complex problems on real simulations required by the ASCI and SSMP mission. The fully integrated heterogeneous, multi-vendor parallel computing environments we are considering pose considerable challenges from system administration to application development to resource management.

Opportunities exist to adapt and develop integrated, scaleable, high-performance software environments based on open source development models, which are highly portable and function effectively across clusters of commodity hardware nodes and interconnect that scale to 4,096-16,384 processors. Software for these highly scalable platforms need to be developed, tested and fielded with scalability, efficient management and ease of use as a primary goal. Because this software base will target, in three to five years, production systems in the 10-to-30 teraFLOP/s range for capacity systems and 100-to-500 teraFLOP/s for production capability systems, it is essential that a new approach to high performance computing software development be attempted by the ASCI program through this PathForward effort.

1.2 Challenges Fielding Large Systems

When fielding the world's largest systems (e.g., Red, Blue Mountain, Blue-Pacific, White, and Q), the ASCI program encounters many problems with the software environment. These "defects" are often difficult to isolate down to root-cause. In addition, the fact that the ASCI systems become classified early in their lifecycle makes the system debugging challenge even greater. In addition, many of the vendor partners providing these platforms do not have systems even within 1/10 the size of the ASCI platforms for debugging and patch verification. All these factors contribute to making the process integrating large ASCI systems take inordinately long periods of time (over a year). Several lessons learned by the program in these efforts have led us to conclude that high-performance computing goals are best served by open source development efforts where the Tri-Laboratory community directly participates in the development and/or testing and/or integration of the technology on platforms at scale.

These conclusions are applicable to the broader high-performance technical computing (HPTC) community for the following reasons:

- HPTC customers require the ability to have operating system and tools source code for root cause analysis and debugging. Normally the HPTC customer sites have the requisite in-house expertise not only to do root-cause fault isolation, but also to formulate and implement bug fixes (that may possibly include re-architecting portions of the solution). With Open Source community development efforts, the Tri-Laboratory can then contribute these fixes back into the community for the betterment of all.
- HPTC customer sites have unique mission requirements that differentiate them from the rest of the scientific community. It is essential that these HPTC sites perform their mission. However, taken as a group, vendors of proprietary solutions indicate that they cannot make a profit providing solutions that span the entire space. Thus HPTC sites are left with two options: live with mediocre solutions that fulfill only part of their mission requirements, or implement major portions of the solution in-house on top of the vendor provided proprietary foundations. Neither of these two options is advantageous for the HPTC or vendor community in the long run. With the Open Source community development model, HPTC sites can contribute their solutions based on mission requirements back to the community because it is based on a common software base and can be contributed back without fear that the development would benefit only one particular vendor offering (and thus inhibit competition in the future and lock the HPTC site into one vendor proprietary foundation). With this wide HPTC scope, the diversity of the debugging environment and developed solutions benefits the entire community.
- Change has been a dramatic feature of the HPTC landscape for many years. However, recent introductions of many disruptive technologies (e.g., killer micros, IA-32 commodity hardware, open source software) have radically increased the rate of change in the industry. As a result, the time span for vendor support of provided hardware and software platforms has been reduced to shorter than the full lifecycle of implemented solutions at HPTC sites. In addition, timescales under which support is withdrawn (from announcement to withdrawal of support) by vendors (measured in quarters) is typically much shorter than the governmental planning period for replacements (measured in years). This mismatch in timescales has led HPTC sites to require Open Source based products as a hedge against "change of support" status.

1.3 Capacity Through Beowulf Clusters

The huge success of the ASCI program to deliver complex modeling capability to the Stockpile Stewardship Program has led to an overwhelming demand for "capacity at the capability level." That is, now that large scale predictive scientific simulation has been adopted as a critical portion of the SSP, there is great demand to run parallel applications for parameter studies, convergence studies and the like at small to medium scale of parallelism. These simulations run at a scale of parallelization that was a "capability run"

for previous generations of ASCI platforms. However, rather than one or even a few capability runs, literally hundreds of delivered teraFLOP/s of computation are required for these production calculations.

The ASCI program is now discussing ways to meet these crushing capacity demands. Although the strategy - and indeed a detailed analysis of the requirements - are still ongoing, one aspect is certain: the solution will be dependent on commodity clusters that are based on open source software (i.e., Beowulf clusters). This is being driven by the fact that Beowulf clusters are 2-10x more cost effective than clusters based on proprietary solutions.

Hence there is tremendous need to provide Open Source HPTC solutions to the ASCI and SSP programs within the next three to five years.

1.4 Leveraging Governmental Funding and Development Efforts

Part of the rational for this RFI is that private companies are interested in return-on-investment (ROI). By providing research and development funding, NNSA hopes to overcome an obstacle many developers within corporations have encountered getting Open Source development projects approved. By providing at least a portion of the non-recurring engineering (NRE) costs, NNSA is reducing the cost to companies for the development of new technologies. In addition, the Tri-Laboratory community offers our resources and development efforts to open source collaborations. A partial list of these efforts is listed in section 5.1. The Tri-Laboratory community will adopt technology developed by successful proposals in our production environments, providing members of the development community direct access and collaborations with a prime target market. These forms of leverage are essential for successful business plans based on technology available in the Open Source.

2 Suggested Open Source Technologies for Acceleration

To meet its goals, ASCI is accelerating open source technologies along multiple dimensions. ASCI clusters stress scale in system administration and software stability. ASCI programming environments stress most software development tools due to scale, software complexity, and compatibility/portability with the ASCI programming models. In addition, almost all ASCI applications are written in-house or under direct contract. This means that code development is a key aspect of overall system usability or time-to-solution. Thirdly, the current ASCI applications efforts have focused on adding physics and engineering capability and scalability. Delivered performance (as measured by time-to-solution as opposed to delivered teraFLOP/s rates) is now becoming a more important aspect of applications programming.

Based on these considerations, the following open source technology areas are suggested for acceleration. We warmly welcome responses to this RFI that are in other technological areas that directly improve the software environment required by ASCI and SSP.

2.1 Technical Areas

The following technical areas are presented as initial starting points for discussion and responses to this RFI. They are summarized below and each is further detailed in an appendix later in the document. When reading these summaries we ask the reader to keep in mind existing open source development efforts, in particular those within the tri-laboratory community. Combinations of these communities could be extremely advantageous to the HPTC community. In addition, there is significant synergy to be had by combining an RFI response to several areas into a single integrated response.

2.1.1 High-Performance Computing Cluster Distribution

Comments on and suggestions for vendor supported Open Source High Performance Computing Cluster distribution(s) (HPCCD) are requested. The HPCCD should address areas of system installation, update, monitoring, debugging, etc., of large clusters.

2.1.2 Resource Management

Comments on and suggestions for open source parallel job resource manager(s) on a Beowulf cluster of up to 4,096 nodes are requested. This resource management solution should be portable to multiple hardware platforms based on Linux OS. The resource management should be able to schedule parallel jobs, control parallel jobs (launch, terminate, status, etc.), account for all work (processes and OS uses of cycles, memory and disk) on the cluster as well as be fault tolerant and highly available.

2.1.3 Open Source System Test/Certification Facility

Comments on, suggestions for, and information on how you would use an Open Source technology system testing facility are requested. The facility will be on a scale representative of the commodity clusters anticipated in the next three to five years. The community will use this facility to develop in an HPTC environment, test at scale, and test candidate bug fixes in released software. In addition, the integrated computing environment that consists of multiple complex software packages (resource managers, parallel file systems, and software provided as part of this effort) must be tested at scale as a complete system. We are interested in testing and benchmarking software to support this activity.

2.1.4 Compilers

Comments on and suggestions for the development (and later support) of Fortran 95/200x and C/C++ compilers are requested. The ASCI applications are extremely stressful to compilers and code development tool chain so close collaborations between the tri-laboratory community and the compiler development team(s) will be necessary. The proposed compilers should be able to compile ASCI applications correctly at high optimization levels on commodity clusters. In addition, OpenMP support for on-node (SMP) type parallelism is needed.

2.1.5 Application Performance Analysis and Correctness Tools

Comments on and suggestions for development (and later support) of tools that can be utilized to gain insight into the correctness and functioning of ASCI applications on large

scale Beowulf clusters is requested. These tools need to deliver performance data in an intuitive way to computational scientists (not computer scientists) and map this information back to the source code statement or basic block level.

2.1.6 Infiniband™ Production Cluster Infrastructure

Comments on and suggestions for software infrastructure for an Infiniband™ (4x or 12x) interconnect for Beowulf production clusters are requested. This should be the minimal and sufficient set of software needed for a complete solution: 1) host bus adapter (HBA) device drivers; 2) switch hardware and/or subnet management; and 3) MPI. This software stack should be implemented on at least one IBA hardware (HBA and federated switch), but would preferably be portable to multiple solutions. The solution should scale to clusters with 4,096 node ports in Clos or fat-tree networking topologies implemented with multiple federated switches.

2.1.7 Visualization Tools and Toolkits

Responses are sought for the development of tools and systems components that support scalable visualization on distributed memory systems with COTS graphics. The datasets generated by ASCI simulations put unique scaling demands on visualization systems, both software and hardware. The proposed software systems should aid in one or more facets of the end-to-end visualization process. Responses for scalable system libraries/APIs, visualization toolkits, applications and their comprehensive integration and packaging are encouraged. Responses that include system software features that enable interactive use and dynamic management of resources within a visualization system or end-to-end visualization environment are also desired.

3 Response Guidelines

If successful, this RFI process may lead to one or more RFPs to solicit specific responses for open source development in the areas chosen by the Tri-Laboratory community (based on responses received to this RFI and other available information). To provide a frame of reference, we anticipate that these development efforts would be no shorter than a major portion of one fiscal year, but less than three to five fiscal years, starting in October 2003 (the beginning of US Government Fiscal Year 2004). It is anticipated that successful future RFP proposals would necessarily be funded at the $1M to $5M level. We believe that RFI responses based on the work of a single graduate student over the summer is probably too small an effort. On the other end of the scale, a combined effort of 100 full-time equivalents over multiple years is probably much too large to be funded via this initiative. This should give some high-level guidance with respect to the length and scope of potentially appropriate projects.

Your response should contain a high level of “content, intention and of course collaboration.” You should describe specific areas of technological interest and capability that could be developed and commercialized by the collaboration. Your response should address the following topics.

3.1 Current Status

What is the current status of the technology components you propose to develop or enhance? What is the current development path and schedule for the technology and what is the anticipated progress within the timeframe of the proposed work without additional ASCI PathForward project support and funding? This area of the response gives the Tri-Laboratory reviewers a description of "the launching point for the project."

3.2 End-State of the Project

If the project were fully funded for the required timescale, what would be the resulting functionality and capability? Describe how this will impact the HPTC community at large and the ASCI Tri-Laboratory community in particular.

3.3 Collaboration

Provide a summary of the proposed collaboration and the background of the key technology providers/developers as it relates to the technology or technologies that form the basis of the project. The ASCI PathForward program is specifically interested in experience, resources, and directly related past experiences. Describe how the project will be managed and in particular how multiple organizations and/or independent contributors will be coordinated. In addition, if there are other models for open source project management that have worked well in the past, please include these suggestions as well. Describe how the source code will be managed. Describe opportunities for and expectations of collaboration with the Tri-Laboratory community in the development project.

In addition, ASCI reserves the right to suggest groupings or combinations of responses, for any possible resulting RFPs or funded collaborations, where appropriate and as long as such suggestions do not violate any proprietary or company confidential information restrictions.

3.4 Accelerated Development

Describe the resources required and timeframe for the project. That is, indicate the human resources required for the success of the project, any access to high-performance clusters hardware environments required (include the scale or size of the resources required) and the timeframe when access would be required at each scaling point. How would the project, if funded, achieve a systematic acceleration in the indicated technology area over that described in the "current status" above? Indicate major milestones in the project during the proposed lifetime of the project.

3.5 Testing and Integration Strategy

It is anticipated that responses to this RFI could be limited to each response element (Appendices A through G) or could potentially contain responses to multiple, but not all, response elements. However, if your organization's proposed response does not include addressing Appendix C, Open Source System Test/Certification Facility, then please provide the following additional information.

What is the testing strategy for the proposed software? How will system test type activities (see Appendix C for the definition of system test) be addressed? What Linux release and or distribution will be the target for the initial development and how will future Linux releases (e.g., Linux Kernel 2.6 generation) be tracked? After this funded development effort concludes how will this software be maintained? This should overlap with the "Technology Productization Strategy."

3.6 Technology Productization Strategy

It is essential that responses not be framed as pure development efforts. Open Source collaborative development efforts that include commercial entities capable of and interested in - but not necessarily currently committed to - long term support of the resulting technology is essential. Responses should have specific productization strategies to turn the technology into supported products by a specific vendor. However, at this RFI stage specific productization commitments are not required. We ask that you indicate how you might in the future commit to commercialization of the developed technology. Indicate any barriers that your collaboration effort might have to making such productization commitments in the future. For work involving the Linux kernel itself, please discuss ways in which the work could be incorporated into the kernel base. If you have experience or capabilities in this area, please provide examples of your experience.

3.7 Project Budget

Provide a rough budget and timeframe estimate for the project. The budget should include manpower estimates either quarterly or yearly with projected costs and include any additional expenses (e.g., travel, equipment, training, etc). Cost sharing by the responder will be a pre-requisite for future successful RFP responses. Indicate the level of any cost sharing anticipated within the collaboration as a fraction of the proposed budget.

3.8 Intellectual Property and Open Source License

The ASCI PathForward program anticipates that intellectual property rights to inventions and copyrights developed under PathForward funding may be retained by the company performing the development, through a process whereby requests for intellectual property waivers are filed with NNSA. Such requests are likely to be granted. Indicate any technology development that you would anticipate needing IP waivers with NNSA. Also indicate the proposed Open Source License or Licenses the collaboration would utilize and why.

3.9 Response Submission

Responses should be submitted electronically to Gary Ward (ward31@llnl.gov) no later than 4:00PM PDT on July 1, 2003. Responses should be divided into multiple sections with information that follows the outline described in sections 3.1-3.8, above and the detailed technical information requested in Appendix A-F. If you are responding to multiple technical areas, then a response should be submitted as a separate electronic

document for each area with an addendum describing the synergy amongst the submissions. Each response should be submitted as a Microsoft Word 2000, or later, compatible document in ".doc" or ".rtf" format.

4 RFI Contact List

Role	Person	Email	Phone	Mail
RFI Contact	Gary Ward	ward31@llnl.gov	Voice: 925-423-5952 Fax: 925-423-8019	Lawrence Livermore National Lab P.O. Box 808, L-550 Livermore, CA 94551-0808
Technical Advisory Committee	Mark Seager	seager@llnl.gov	Voice: 925-423-3141 Fax: 925-423-8719	Lawrence Livermore National Lab P.O. Box 808, L-060 Livermore, CA 94551-0808
Technical Advisory Committee	Neil Pundit	pundit@sandia.gov	Voice: 505-845-7601 Fax: 505-845-7442	Scalable Computing Systems, 9223 Sandia National Laboratories POBOX 5800, MS 1110 Albuquerque, NM 87185-1110
Technical Advisory Committee	Mike Koszykowski	mikek@sandia.gov	Voice: 925-294-3257 Fax: 925-294-9915	Sandia National Laboratories 7011 East Ave, MS 9915 Livermore, CA 94551-9900
Technical Advisory Committee	Jeff Brown	jeffb@lanl.gov	Voice: 505-665-4655 Fax: 505-665-0120	Los Alamos National Laboratory POBOX 1663 Los Alamos, NM 87545

5 Additional Information

5.1 ASCI Open Source Software Development Efforts

Lawrence Livermore National Laboratory Linux development efforts

http://www.llnl.gov/linux/

Sandia National Laboratories CPLANT project

http://www.cs.sandia.gov/cplant/

Los Alamos National Laboratory

http://www.public.lanl.gov/cluster/index.html
http://www.linuxbios.org/

5.2 Additional Websites

- US Department of Energy www.doe.gov
- National Nuclear Security Administration www.nnsa.gov
- Los Alamos National Laboratory www.lanl.gov
- Lawrence Livermore National Laboratory www.llnl.gov
- Sandia National Laboratory www.sandia.gov
- ASCI www.llnl.gov/asci
- ASCI Platforms www.llnl.gov/asci/platforms
- ASCI PathForward www.llnl.gov/asci-pathforward
- ASCI Benchmarks www.llnl.gov/asci/applications, www.llnl.gov/asci/purple/benchmarks/
- Open Source Licenses www.opensource.org/licenses/index.php
- Open Source License Law Resource Center www.denniskennedy.com/opensourcelaw.htm

6 Appendix A – High-Performance Computing Cluster Distribution

6.1 Background/Requirements

Mainstream Linux distributions are not well suited to the requirements and rigors of HPTC computing. Limitations exist in the kernel proper, and support for the management of large clusters is lacking. Additionally, the different Labs have (sometimes radically) different approaches to building these large HPTC Linux clusters, including use of lightweight kernels, single namespaces, LinuxBIOS among others. This complicates most attempts to agree on a standard Linux cluster distribution. Yet, some common objectives do exist. We have itemized some of these below and seek proposals that address these areas of interest.

6.2 Current state-of-the-art

As mentioned, the three Labs have taken different approaches to deploying large Linux clusters. LANL Clustermatic, LLNL CHAOS and Sandia Cplant each include innovative technology that should be evaluated by the respondents. Furthermore, the open source community project Open Source Cluster Application Resources (OSCAR) contains significant cluster deployment technology and should be evaluated as a cluster deployment methodology. Wherever possible, proposed solutions should have benefit to the entire TriLab community.

6.3 Areas for greatest impact

Responses are requested for a vendor-supported Open Source High Performance Computing Cluster Distribution (HPCCD). Ideally these HPC enhancements should be incorporated into the vendor's mainstream Linux distribution. If a separate distribution is justified, it should be closely synchronized to the mainstream distribution, with timely bug fixes and an aggressive release cycle. Requested enhancements fall into two broad categories: enhancements to the Linux Kernel to support HPC and scalable cluster management solutions.

Enhancements to the Linux kernel to support HPC computing environments are requested. These should include, but are not limited to, the following.

- Large file system support - There are a number of areas in the Linux kernel, device drivers and utilities that limit file system sizes to 2TB. This limit should be raised to 10PB.
- High Speed SCSI layer support - Significant performance limitations are present in the SCSI layer of the Linux kernel. Enhancements should be made to deliver an aggregate SCSI I/O performance of at least 350 MB/s from a single dual 2.8 GHz Xeon server with at least two, but at most four, attached SCSI devices.
- Quadrics Elan3 and Elan4 device driver support - The Open Source Quadrics QsNet Elan3 and Elan4 device drivers should be integrated into the Linux kernel.

- Support for handling ECC memory errors – The Linux kernel should detect uncorrectable memory errors in real time and optionally crash the node (as a configurable option).
- Infiniband support – Kernel support for Infiniband should be supported (see Appendix F – Infiniband™ Production Cluster Infrastructure).
- Advanced architectures – The Linux kernel should provide support for AMD Opteron (x86-64) and Intel Itanium (IA64) architectures.
- Crash dump solution – A mechanism is required for capturing memory state after a kernel panic and for analyzing the resulting crash dump.
- IA32 Large page support – Support is required for page sizes > 4K, or for variable length pages is required to reduce the TLB impact of codes with large, fragmented working sets.
- IA32 Large memory support - Remove barriers (size of VM data structures) to support configuring IA32 systems with greater than about 8G physical memory.

Tools are desired to aid in the installation and management of large HPC Linux clusters. Solutions should be scalable to clusters of 4096 nodes. We are particularly interested in the following areas:

- Cluster installation tool – For disk-full clusters, tools are needed to install the Linux distribution on cluster nodes and perform post-install customizations. Open Source Cluster Application Resources (OSCAR) is an example of a greatly simplified cluster deployment mechanism that enables one to perform pre- and post-cluster install configuration and customization. This open source package is available from www.OpenClusterGroup.org/OSCAR. This tool should be highly scalable and able to install a 4,096 node cluster in under one hour.
- Diskless client support – Strategy and associated tools are needed to efficiently build and manage diskless NFS root cluster nodes. Strategy should include, but is not limited to: kernel support for context dependent symbolic links, hooks to allow scripts embedded in RPM scripts to execute appropriately in a diskless context, repackaging of standard RPM's to operate in a diskless environment, and support for a no-swap kernel configuration that reserves memory for critical system tasks. An example of this open source technology is the thin-OSCAR project available from http://thin-oscar.ccs.usherbrooke.ca/
- Bproc support – tools to enhance the management of bproc-based clusters, for example, providing for logical partitioning of bproc clusters into multiple domains and merging of these partitions back into one cluster. We also would be interested in supporting integration of bproc into a standard kernel distribution.
- System monitoring – Tools are needed to efficiently gather Reliability, Availability and Serviceability (RAS) data from cluster nodes and display this data in a concise and meaningful graphical format. Tool should be configurable to gather any arbitrary statistics from the nodes and to define threshold values for error conditions. Data gathering shall be performed efficiently without significant overhead on node or network resources.
- Console management – A tool is required to manage serial console output from cluster nodes. This tool should be able to capture console output and consolidate for centralized access, provide real-time access to the serial console of individual nodes

as well as the ability to issue commands across the serial console of multiple nodes in parallel.

- Network Console – There are several network console systems extant. Both kernel and user tools are needed to converge on a single console system and to ensure that it has the capabilities need to completely eliminate serial consoles where that is desired.
- Power management – A tool is required to perform centralized power management (power on, power off, power status) of cluster nodes. This tool should be able to operate on 4,096 nodes in parallel and complete an power on, power off or power status operation under five seconds for 4,096 nodes.
- User level and system level parallel cluster tools that provide administrators and users with a Single System illusion (SSi) view of the cluster are required in order to simplify operations across cluster nodes. This tool should be both scriptable and available from the command line. It should facilitate both the execution of tasks and the movement of files; both scatter and gather operations, on all or any subset of cluster nodes. One example of such an open source tool is Cluster Command & Control (C3) available at http://www.csm.ornl.gov/ClusterPowerTools/.
- An attempt should be made to eliminate the necessity of a serial console or other specialized non-commodity (COTS) hardware for the purpose of providing node console management and network management tasks. Instead, COTS based technology such as the Intelligent Platform Management Interface (IPMI) should be leveraged to provide a uniform software based interface to node management across hardware vendors. Hardware vendors should then be encouraged to provide the low-level plug-in modules to facilitate the direct control of their hardware from this middleware layer (note: no additional work will be required of a vendor if they support standard IPMI.) An upper level console GUI should be provided for direct user level interaction with the systems. The console should provide management capabilities from cluster wide to subsets of cluster nodes.

6.4 Area specific requests for information, if needed

Indicate how the development activity will track changes to Linux and subsystems associated with a common Linux distribution (e.g., United Linux or RedHat).

Indicate how the development activity will integrate with packages developed in Appendix B through Appendix G.

7 Appendix B – Resource Management

7.1 Background/Requirements

Resource management in this context means managing pending work (i.e. jobs), deciding when and where to initiate those jobs, dispatching those jobs, and managing them throughout their lifetime.

7.2 Current state-of-the-art

The Portable Batch System (PBS, www.openpbs.org) is a flexible batch queuing and workload management system originally developed by Veridian Systems for NASA. It

operates on networked, multi-platform UNIX environments, including heterogeneous clusters of workstations, supercomputers, and massively parallel systems. PBS is available in both an open-source (GPL) version called *OpenPBS*, a pseudo open source version requiring registration called *PBS*, and a commercial version called *PBS Pro*. The current owner of PBS Pro is Altair Grid Technologies (http://www.altair.com/). Open PBS is widely used and is part of the OSCAR (http://www.openclustergroup.org/) cluster management suite along with the Maui Scheduler (http://mauischeduler.sourceforge.net/).

PBS supports sophisticated scheduling logic (via Maui). It spawns daemons on each machine to shepherd the job's tasks. It provides an interface for administrators to easily interface their own scheduling modules. PBS can support long delays in file staging (in and out) with retry. Host authentication is provided by checking port numbers (low ports are only available to user root). A credential service is used for user authentication.

Major problems with PBS (as of late 2001) include:

- PBS is single-threaded. Node failures result in server performance problems. Large systems can also suffer from poor performance.
- Bugs are fixed in the *PBS Pro* system first. The *OpenPBS* version does not get the bug fix until *PBS Pro* undergoes two major revisions, which makes support difficult.

PBS has a simple scheduling algorithm, so it is typically used in conjunction with the Maui Scheduler. Maui is not a full-service resource manager, but a very powerful and popular system to direct how pending jobs are scheduled by an underlying resource manager such as PBS or LoadLeveler.

Other Resource Managers include:

- Quadrics' Resource Management System (RMS, www.quadrics.com) works very well, but only if you use a Quadrics interconnect.
- Platform Computing's Load Sharing Facility (LSF, www.platform.com) also works very well and supports many systems, but is proprietary and difficult to extend.
- IBM's LoadLeveler

  (www-1.ibm.com/servers/eserver/pseries/library/sp_books/loadleveler.html) supports a limited variety of computers and has performance problems.
- Platform Computing's Load Sharing Facility (LSF, www.platform.com/) has good functionality.
- Sun Grid Engine (SGE, www.sun.com/gridware) seems to work quite well and supports many systems, but is a proprietary binary release (at present, used to be open source).
- SciDAC Scalable Systems project (www.scidac.org/ScalableSystems) is developing a resource manager. Early design work looks fine, but the timeframe for deployment of a system with the required fault-tolerance and scalability is

scheduled for June 2006. Some of their early design decisions could impact system scalability.

- Condor (www.cs.wisc.edu/condor) is really designed to harvest unused resources on heterogeneous workstations. While it has some really nice features, it really is targeting a different problem.
- MauiME (http://www.sourceforge.net/projects/mauischeduler) looks quite attractive in terms of functionality and is open source. MauiME is new software (release in late 2000). Note that MauiME and the Maui Scheduler are distinct software products. MauiME is a complete resource manager and scheduler. It is written in Java and may lack the desired level of scalability (at least according to the developers when our requirements were discussed).
- BJS is a scheduler designed for bproc clusters. BJS supports policy modules as plug-ins, rather than external processes. Since BJS uses bproc it is able to eliminate the node daemons required with, e.g., Condor and PBS. BJS runs as a single process. BJS will support the future MPI-2 requirements for dynamic job size (jobs can increase or decrease the number of nodes they use during a run). BJS has proven itself in clusters of order 128 nodes, but its performance on 1024-node clusters has not yet been proven. BJS does not yet provide as complete a scheduling solution as LSF.

7.2.1 Resource Management for Production Linux Clusters

As part of the migration to Linux clusters, LLNL reviewed the above resource managers and found that only Quadrics RMS had the features and performance that LLNL required. However, RMS is proprietary and only supports the Quadrics QsNet interconnect. To overcome these limitations, LLNL set out to develop a Simple Linux Utility for Resource Management (SLURM, www.llnl.gov/linux/slurm) to satisfy our requirements beginning in late 2001. The design and early development was done jointly with Linux NetworX (www.lnxi.com). As of late February 2003, SLURM is deployed on some development systems and plan to deploy on large Linux production systems shortly. Some key features of SLURM include:

- Open source (GPL)
- Portable: Written in C with GNU *autoconf* configuration engine and designed for portability. Currently supports TCP/IP and Quadrics Elan3 interconnects with more planned for the next release including Blue Gene, InfiniBand and Myrinet (including support for topographic resource scheduling constraints).
- Highly scalable (performance comparable to RMS and 80 times faster than LoadLeveler). Very good performance demonstrated to 950 nodes.
- Highly fault tolerance including automatic fail-over SLURM controller.
- Security based upon Pluggable Authentication Modules (PAM) using *authd* (http://www.theether.org/authd).
- System administrator friendly. Very large clusters can typically be configured in only a few lines.

7.2.2 Research on Single System Image Linux Cluster Resource Management

LANL is currently doing research on Linux cluster resource management leveraging the bproc "Single Process Space" system. Bproc is one component in LANL's development of a cluster management system that provides Single System Image to applications.

Currently, LANL's research path includes using a bproc-enabled version of LSF from Platform Computing on the Pink cluster. In the long term, however, we would be interested in research responses that leverage the BJS scheduler or extensions to SLURM that incorporate a bproc single system image cluster as a node in a meta-cluster. Other Open Source schedulers that leverage bproc would also be of interest, as long as they have at minimum the capabilities that BJS provides.

7.2.3 Areas for Greatest Impact for Production Clusters

Responses that leverage the SLURM effort for production-oriented schedulers are encouraged. In particular, Open Source organizations interested in SLURM for harnessing large Linux clusters could add support for additional interconnects, operating systems, and/or authentication mechanisms and meta-clusters with bproc single system image nodes. These changes would be incorporated into the SLURM code base. In addition, responses targeted to add support for checkpoint/restart in SLURM when the Linux operating system supports this facility would be of value. Finally, responses targeted at integrating SLURM into SciDAC would be viewed positively from both the DOE/NNSA and DOE/Office of Science.

Other responses for Open Source resource management that are independent of SLURM are also encouraged. However, such responses would need to have a large development community and committed vendor support in order to be deemed of sufficient value to warrant Tri-Laboratory follow-up.

7.2.4 Areas for Greatest Impact for Resource Management Research

Responses that leverage the bproc system for innovative single system image cluster resource management research are encouraged. BJS is a possible approach to such a response. Another possible approach is to modify SLURM to schedule multiple jobs by tracking resource consumption (e.g., CPUs and memory) within a bproc cluster as a node in a meta-cluster. Whatever approach is proposed, that resource scheduler should make full use of bproc capabilities for managing resources within the bproc single system image.

7.3 Area specific requests for information, if needed

Indicate how the development activity will track changes to Linux and subsystems associated with a common Linux distribution (e.g., United Linux or RedHat).

Indicate how the development activity will integrate with compilers developed in Appendix D and application performance analysis and correctness tools developed in Appendix E.

Indicate how the development activity will mange the graphics consumable resources developed in Appendix G.

8 Appendix C – Open Source System Test/Certification Facility

Responses are requested on responder's needs and potential use for a full-scale development, testing and certification facility. In addition, information on responder's proposed open source software development and testing methods, that could utilize a full-scale development, testing and certification facility, is also requested.

We are also seeking software solutions to accomplish comprehensive, full scale testing and certification of complex, multiple package software environments.

8.1 Background/Requirements

Testing proposed total software environments at scale and in concert with other aspects of the complete computing environment is essential for successfully fielding large production Linux clusters. If this system test and certification effort is successful, all software delivered in response to future Tri-Laboratory Linux cluster RFPs is required to have been deployed and tested on a real system at scale (e.g., a test and certification facility).

The open source community seems to be able to develop kernels, packages (e.g., Perl and Apache web server) and groups of utilities (e.g., GNU compilers and GNU text utilities) and test these independently as "packages" in a quite effective manor. In formal software development terms, these groups appear to do function verification and component testing very well. In large part, the huge success of Linux is due to the quality of these development *and testing* efforts. However, the formal "system testing" which is the testing of an entire solution (e.g., various Linux distributions) seems to be cursory at best. The profuse patch stream for most Linux distributions can be attributed to this.

Unfortunately, this state of affairs places a great deal of burdensome "value added testing" to field real production quality Linux clusters. This effort, because it is not coordinated, must be duplicated by every group fielding clusters. It raises the bar substantially on the skill level required to do cluster integration and makes the learning curve much steeper. This in turn slows the adoption of large (>1,000 nodes) Linux clusters by a broader community and raises the risk of building really huge (>4,096 nodes) Linux clusters to unacceptable levels.

Our collective experience has been that most cluster vendors and open source providers, do not have computing systems large enough to test software at sufficient scale, nor do they have computing environments that resemble those in use within Tri-Laboratory community.

The goal of the system test and certification facility, and the software supporting it, is to provide a mechanism for open source contributors to develop, test, and debug both at scale and in an appropriate HPTC computing environment. Additionally, cluster providers can use this "certified" software in their solutions with minimal additional testing for unique hardware or turn-key solutions.

The facility will also provide an environment suitable for testing performance, scalability, robustness, and interoperability of a software release. We are interested in open source software to support the benchmarking and certification efforts including test suites, benchmarking tools, and code harnesses. While approaches that are specific to a particular distribution or release are of interest, we would prefer solutions that are applicable to the different variations of systems that exist across the Tri-Laboratories.

8.2 Current state-of-the-art

The final details of the testing and debugging facility to be available to the open source community will depend to some degree on the responses to this RFI, as well as on available resources.

However, current systems are described here in order to provide suitable background for responses.

- DOE's Office of Science operates an open (i.e. not classified) testbed system for open source development and testing at Argonne National Laboratory called "Chiba City". Chiba City is several years old and limited to 256 physical nodes, but is available to participants in this program. Information about Chiba City can be found at http://www.mcs.anl.gov/chiba/. Chiba City has the following features that are relevant to this RFI:
    - A "standard" Linux OS environment is provided on all computing nodes, but users of the cluster can provide their own complete OS for the nodes, if, for example, they are testing an OS distribution.
    - If necessary, users of the system can be granted root access on the nodes, can reboot nodes remotely, and access node consoles.
    - It is possible to reserve the system for exclusive use for testing for an extended period of time.
    - Larger testbeds in a similar vein are under consideration for the future. If created, these will include the ability to test on "virtual" nodes, allowing scalability tests of potentially eight thousand OS instantiations.
- Several NNSA clusters may also be available for testing purposes on a limited basis, including Pink at LANL (http://www.lanl.gov/projects/pink/) and MCR system at LLNL (http://www.llnl.gov/linux/mcr/).

A large number of solutions for certification and testing of software exist, the bulk of which are proprietary and/or of very limited scope. A few general open source solutions exist, some of which are listed here to provide context:

- Many current Linux distributions, such as RedHat and SuSE, and HPC software collections, such as OSCAR and NPACI Rocks, carry out internal testing and certification in order to ensure consistency within a distribution.
- The SciDAC Scalable Systems project (http://www.scidac.org/ScalableSystems) is developing a standardized system software suite for large-scale clusters. Testing, validation, and integration of complex system software components are important parts of this effort. For example, the project is developing a program called

APITEST, which is meant to allow easier testing of the networked components through their interfaces. This allows the developer to make very specific tests to validate functionality and specific paths in the runtime system without resorting to intensive coding for testing.

8.3 Areas for greatest impact

Having systematic testing and certification software tools will be of benefit to all areas of software development. But the areas of particular interest are:

- Tools and mechanisms to test an entire set of software working in conjunction, i.e. something that can test a resource manager, parallel file system, compilers and libraries, and so on, all together.
- Tests related to the software packages requested elsewhere in this RFI.
- Software that can test specific functions of a program and then capture the state of that test. The environments that the software will ultimately be deployed in are likely to be highly variable, thus a certification of success of some program may need to include information such as the version of Linux, types of libraries, size of memory, versions of various key programs, and so on. This information could then be recorded along with the results of the tests for reference later when looking into a particular problem.
- General certification and testing harnesses that are similar in spirit to those used by vendors to validate the consistency of their software releases. These are used both for open source distributions and proprietary solutions.

While approaches that are specific to a particular distribution or release are of interest, we would prefer solutions that are applicable to the different systems that exist across the Tri-Laboratories.

8.4 Area specific requests for information

We seek responses to the following questions as appropriate:

1. If you are proposing a solution for software package and environment verification and testing, describe that proposition in detail.
2. What are your hardware and software requirements for development and testing of your solution at scale?

   Please discuss how large of a testbed facility you would need, and what specific configuration requirements (e.g. global and/or parallel file system, local file system or not, resource management environment, etc) you would need to effectively test. Describe how you will test and validate your solution, including descriptions of specific tests and test suites you use. Describe the access requirements you have such as number of accounts, cluster usage pattern (dedicated, intermixed with other user, debug shots, etc) and durations of access.

3. As noted in section 3.5 above, please describe your software release strategy and any mechanisms you have for insuring and testing interoperability with other software.

Indicate how the development activity will track changes to Linux and subsystems associated with a common Linux distribution (e.g., United Linux or RedHat).

Indicate how the test and certification facility can be utilized by the other development activities envisioned by this RFI.

9 Appendix D – Compilers

9.1 Background/Requirements

The NNSA Tri-Laboratory HPC community has made a significant investment in developing complex simulation capabilities. The major applications use a variety of software development languages, including: Fortran 77, Fortran 9x, C, C++, Perl, Python, Java, and Tcl. The scripting languages are all mature and the requirements on them are relatively light. The compiled languages are inevitably bearing the brunt of the demands that HPC applications make. C and C++ are currently in high demand in the commercial sector, although certain specific technologies required by the HPC community are a low priority commercially. Fortran is still the only major language that assumes no aliasing so that it may apply extra optimizations. The Fortran language standard requires that the user indicate when aliasing exists. C++ rejected the keyword *restrict*, which means there is no standard way to tell the compiler aliasing doesn't exist. C includes the *restrict* keyword allowing some very sophisticated optimizations not possible in other languages. The commercial developer user base for Fortran is essentially non-existent.

There is a large legacy code base of Fortran 77 code. One popular open-source compiler, g77, cannot even begin to compile any of this legacy code due to missing support for Cray-style pointers and many other legacy extensions. Another popular open-source compiler, Open64, cannot compile any of this code because it lacks support for the required platforms. This legacy code base needs to be maintained and ported to new platforms. There are several code efforts actively developing Fortran 77/9x software.

We are seeing a remarkable decline in the Fortran user community both inside and outside of the general HPC community. Many HPC community codes are turning to C/C++, but Fortran is still very important in the NNSA Tri-Laboratory HPC community.

A possible reason behind the decline in the Fortran user base is the lack of a standard-compliant, open-source compiler. There is no single cross-platform, standard-compliant open-source compiler solution.

The open-source aspect would get a compiler into the hands of universities at no cost. This would serve to increase the Fortran programmer base, which will, at the very least, provide a base from which to draw Fortran programmers. At best, it will make the language more popular which would in turn increase the likelihood of better quality Fortran compilers and tools. Furthermore, the open-source aspect would allow other vendors to use the front-end with their back-ends.

A possible outgrowth of this work may be an open and eXtensible IR (Intermediate Representation) – XIR. This XIR would allow vendors to differentiate themselves on the quality of the optimized code they produce for a given platform and the analytical tools they provide for debugging and improving the quality of the applications. Properly implemented, this layer would be the ideal place for 3rd party optimizations and software tools. Tools developed at this layer would be language independent and yet still have access to all the high-level abstractions and detailed information possessed in the original programming language.

The one tool that touches every piece of software developed is the compiler. This critical area needs improvements that lead to more reliable, faster and more available software tools to meet the scientific missions of DOE. Even the kernel, at the core of the OS, benefits from improvements in compiler technology like optimizations for register coloring and loop unrolling. This is an often-neglected area that can have as dramatic an effect on performance as developments in hardware.

The successful project will, at a minimum, lead to the following:

- A test suite will be used to verify performance and validate correctness of the work. Later, this test suite may be used by the moderator of the open source code repository to verify that changes submitted by someone work. Of particular importance are deeply nested modules in Fortran and templates in C++.
- An open-source license selected for the Fortran 95 and Fortran 200x front-ends that allows those that use them to choose whether or not they will distribute their source code with their derivative works. That is, the producers of a derivative work using these front-ends may bundle and ship the front-ends, and their source code, with the derivative work, but ship proprietary derivative work as binary.
- Preference given to a solution where the open-source license for the C/C++ front-end is the same as for the Fortran front-ends and allows choice in the matter of distributing the source code of derivative works that use the C/C++ front-end.
- C (ISO/IEC 9899:1999, published 1999-12-01) and C++ (ISO/IEC 14882:1998, published 1998-09-01) front-end with compile times no worse than 10% slower than a selected reference standard for the platform and OpenMP 2.0 support. This front-end shall be tied to back-ends that can produce optimized executable code for at least Tru64/Alpha, AIX/Power5, Linux/IA-32, and Linux/Opteron. Additional back-end support is desired for IRIX64/MIPS R10K, AIX/Power3 and Power4, and Linux/IA-64, but is not required.
- Fortran 95 (ISO/IEC 1539-1:1997) front-end with compile times no worse than 10% slower than a selected reference standard for the platform, interoperability with C functions and types from the C compiler specified in the paragraph on the C (ISO/IEC 9899:1999, published 1999-12-01) front-end, OpenMP 2.0 support, and a set of negotiated extensions - Cray Pointers, VAX, etc. This front-end shall be tied to back-ends that can produce optimized executable code for at least Tru64/Alpha, AIX/Power5, Linux/IA-32, and Linux/Opteron. Additional back-end support is desired for IRIX64/MIPS R10K, AIX/Power3 and Power4, and Linux/IA-64, but is not required.

- Fortran 200x (ISO/IEC JTC1/SC22/WG5 N1497) front-end with compile times no worse than 10% slower than a selected reference standard for the platform, interoperability with C++ objects from the C++ compiler specified in the paragraph on the C++ (ISO/IEC 14882:1998, published 1998-09-01) front-end, and OpenMP 2.0 support. This front-end shall be tied to back-ends that can produce optimized executable code for at least Tru64/Alpha, AIX/Power5, Linux/IA-32, and Linux/Opteron. Additional back-end support is desired for IRIX64/MIPS R10K, AIX/Power3 and Power4, and Linux/IA-64, but is not required.
- Preference given to a solution that provides the widest selection of back-end support.

Each optimized back-end must produce executable code for a negotiated suite of applications that performs no worse than 10% slower than the native compiler for that platform. The executable code must be able to be debugged back to the original source code. This allows the option of a back-end that emits a standard-compliant, high-level language such as C (ISO/IEC 9899:1999, published 1999-12-01) and using a compiler for that language on the platform to produce the executable code. If this implementation is chosen it must meet the performance and debugging requirements.

These requirements produce a path forward in compilers that is beneficial for the Tri-Laboratory HPC community because:

- A cross-platform compiler suite for the primary programming languages used for the codes with competitive performance is produced. This will limit the current practice of special programming to compensate for differences in platform-specific compilers, thus minimizing development costs.
- A free compiler with a wide variety of target platforms gets into the hands of universities so that more Fortran expertise can be promoted.
- Front-ends that can syntactically parse complex codes are made available to compiler and tool providers to use. This produces a wider set of compilers and tools that will be known to be able to syntactically parse these codes. Currently code teams cannot use certain tools because they cannot syntactically parse the code and consequently this HPC community cannot benefit from their value-added functionality.
- As compiler and tool providers make use of the front-ends they will add functionality and fix problems that will help everyone and are obligated to share those front-end fixes with the other users. This HPC community will be able to use better products.

9.2 Current state-of-the-art

There are several open-source projects involving Fortran. SGI donated its Open64 compiler suite, a small g95 effort is underway, and the old g77 code base still exists. Each of these existing projects has limitations.

The SGI code base is an important contribution to the open-source community. Unfortunately, the original compiler suite was based in part upon proprietary technologies that did not belong to SGI. So the contributed code base has been stripped of

these technologies. For example, the C++ parser from EDG was replaced with the old GCC parser that LANL funded Code Sourcery to replace in GCC. Also, this product runs on a limited number of hardware platforms and is written using pre-standard C++, which would require significant effort to modernize. Consequently, finding a compiler to compile the compiler may become an issue. This effort is hampered by its choice of the GPL, which limits the ability of commercial companies to supply a custom, proprietary back-end to the front-end or to use the front-end in a tool.

The g95 effort is small and has proceeded slowly. This effort is hampered by its choice of the GPL, which limits the ability of commercial companies to supply a custom, proprietary back-end to the front-end or to use the front-end in a tool. The g95 effort does not currently generate code since it is not yet tied into the GNU back-ends.

The g77 code base has the same licensing issues as g95, but it also suffers from being ancient technology, which would essentially require a complete rewrite to be able to modernize its capabilities. This effort is based upon ATT's *f2c* source-to-source translator.

9.3 Areas for greatest impact

The NNSA Tri-Laboratory HPC community has a very large investment in Fortran and needs to ensure the existence of a quality Fortran compiler for the platforms that are supported. This compiler will serve as the risk-mitigation backup to the platform-specific compiler. If it performs better it could be the primary compiler for a given platform. The open-source aspect would get a compiler into the hands of universities at no cost. This serves to increase the Fortran programmer base, which will, at the very least, provide a base from which to draw Fortran programmers. At best it will make the language more popular and increase the likelihood of better quality Fortran compilers and tools. Furthermore, the open-source aspect would allow other vendors to use the front-end with their back-ends.

The NNSA Tri-Laboratory HPC community also has a significant investment in C/C++. This compiler will serve as the risk-mitigation backup to the platform-specific compiler. If it performs better it could be the primary compiler for a given platform. The open-source aspect would allow other vendors to use the front-end with their back-ends.

The possible outgrowth of an open XIR would allow vendors to differentiate themselves on the quality of the optimized code they produce for a given platform and the analytical tools they provide for debugging and improving the quality of the applications.

Any improvements to the compilers potentially affect the performance of every piece of software from the interpreters, to the kernels, to the end user applications software.

9.4 Area specific requests for information, if needed

In addition to answering the questions in section 3 (proposal preparation), the following area specific information should be included.

The compiler needs to track any changes to the operating system (e.g., thread model changes in Linux OS 2.6) and tools (e.g., binary format changes). Please describe your approach for tracking these changes.

Any potential development plan in this area should address the following issues:

How many hardware platforms can the solution support? What are the barriers to cross-language development inherent in the solution? What are the advantages of the approach vis-à-vis cross-language support? How long would it take to develop a Fortran 200x solution useful in this HPC community's scenarios? What, if any, intermediate states might be of use to the development community (e.g. Fortran 95 after one year, Fortran 200x after two years)? What licensing issues would the approach entail? What would be the minimum product necessary to jumpstart active development in the open-source community? What successful programs have been led by the respondent? What are the extensions necessary to support legacy codes? What will be your open source license agreement? How extensible is the solution, e.g. could it easily support language extensions such as HPF or Co-Array Fortran?

Indicate how the development activity will integrate with application performance analysis and correctness tools developed in Appendix E.

10 Appendix E – Application Performance Analysis and Correctness Tools

Responses for the development (and later support) of tools that can be utilized to gain insight into the correctness and functioning of ASCI applications on large scale Beowulf clusters are requested. These tools need to deliver results in an intuitive way targeted primarily to computational scientists (not computer scientists) and map this information back to the source code statement or basic block level.

10.1 Background/Requirements

Detailed tool requirements are too long to list here. Previous ASCI platform RFP Statements of Work are a source of detailed requirements (cf. www.llnl.gov/asci/purple/). The following are general high-level requirements:

- Tools are needed for Fortran95, C, and C++ and mix of these languages – including OpenMP directives.
- Target applications may have thousands of procedures and hundreds of thousands of lines of code.
- Tools and analysis mechanisms need to scale to the area of 10,000 processors.
- Launch of the tools must be fast and scalable to systems with 10,000 processors. Launch and interactivity should be measured in seconds and 10's of seconds rather than minutes and 10's of minutes when dealing with large codes and high processor counts, especially for routine interactions.
- Techniques requiring recompilation should be minimized, especially complete recompilation – since this may require many hours.
- Ability to attach to running applications is desired.

- If trace data is generated, mechanisms to control amount of output and dynamically and repeatedly activate and deactivate tracing are critical.
- Parallel models currently in use are primarily MPI with some OpenMP usage within an MPI task. The pthreads interface is also used.
- The default mode for tools should be easy to use and should not have a huge learning curve for computational scientists (i.e., designed with the end user in mind). Results of code analysis should be easily understandable by normal humans who think more about Physics and Engineering rather than the details of computers and compilers. Expert features for the specialist or computer scientist, who needs to track an especially difficult problem, but the emphasis is on tools for more general use.

10.2 Current state-of-the-art

Current open source parallel application analysis tools are limited. For example, most Linux specific tools address a single processor only. If they can be used in parallel, they generate hundreds and thousands of files that are, in effect, useless to a computational scientist. Following are a set of tools that represent the current state of the art on ASCI systems, along with some of their limitations. Unless otherwise indicated, these tools are commercially available.

- **MPI Profiling:** Vampir, mpiP [LLNL]
- **Hardware Counter access and instruction profiling:** PAPI library [UTK]
- **CPU Profiling:** VGV (limited portability) and TAU [U. of Oregon] for instrumented profiling, VProf [Sandia] for statistical profiling
- **Debugging:** Totalview and Guard
- **Memory Correctness:** Purify (limited portability, scalability issues), Zerofault (limited portability, scalability issues), valgrind [open source] (limited portability, scalability, requires instruction simulator), Insure (scalability issues)
- **Multi-threading Correctness:** Assure (limited portability, scalability issues)
- **Memory performance, especially for NUMA systems:** TAU (limited information)
- **Memory performance prediction:** MetaSim [SDSC] (requires commercial software--cycle accurate simulator)
- **MPI performance prediction:** Dimemas (limited portability, scalability)

10.3 Areas for greatest impact

Following are areas of potential impact of OSS work on tools in order of highest to lowest. To be of interest, these tools must address systems of high scale and executables with mixed standard languages. More than one area might be addressed within a single tool.

- Memory Correctness
- Memory Performance
- CPU Performance
- Multi-threading correctness
- Debuggers
- MPI Performance

Several lower level requirements could have a big impact on existing (and future) tools:

- Instrumentation package, including source browser to allow easier tool implementation, (e.g. ToolGear at LLNL)
- stack unwind library
- a better binutils
- a binary editing utility
- continued porting and functionality in the PAPI library
- switch performance interface

10.4 Area specific requests for information

Any response to this RFI should include information about the following:

- Which systems are targeted by the tool (include any restrictions on binary formats)?
- Which languages are supported?
- What level of scalability is targeted?

Indicate how the development activity will track changes to Linux and subsystems associated with a common Linux distribution (e.g., United Linux or RedHat).

Indicate how the development activity will integrate with compilers developed in Appendix D.

11 Appendix F – Infiniband™ Production Cluster Infrastructure

11.1 Background/Requirements

Initial 10 Gb/s Infiniband (IB) 4X hardware and software have been measured as capable of exploiting in excess 800 MB/s of link bandwidth and less than 7.0 μs latencies for MPI applications. This performance level achieved by initial implementations indicates potentially significant benefits for utilizing IB in capacity computing environments. The Infiniband software infrastructure for HPC includes Linux device drivers, a subnet manager, and MPI. HPC places stringent requirements on software including high

performance and scalability. DOE supercomputers are using 1,000's of processors today, and are expected to employ 10,000's of processors in the foreseeable future.

To meet these requirements the Infiniband subnet manager should be redesigned to be highly scalable and fast (order of seconds) with respect to fabric discovery, address assignment, routing, performance and connection management support. It is thought that an Infiniband fabric simulator would provide the necessary environment to test the scalability of the subnet manager towards future large-scale platforms. The Infiniband kernel network driver stack should be an integral single HPC Linux driver supplying all the Infiniband features necessary to HPC while using a minimal memory footprint and providing high performance. The driver should be Linux distribution independent and portable across different Linux kernel versions. The ability to explore alternative fabric topologies would permit maximization of the performance to price ratio.

Alternative topologies can be most cost effectively explored with a scalable Infiniband fabric simulator that allows optimization of fabric initialization, routing for relatively constant cluster configurations and may even characterize computational performance from MPI statistics would be valuable. IB overall fabric management needs extensive exploration to define for HPC the roles of subnet, communication, performance, baseboard and application management. Latency reduction in the Infiniband Verbs layer, for both polling and interrupt driven modes, selection of consistent interfaces and tuning for increasing the message passing bandwidth will induce performance gains in MPI. MPI performance for collective communications and I/O will be further enhanced by the availability of IB RDMA, multicast and atomic operations, and transport service levels and types. These features must be implemented and tuned in IB and MPI libraries must be modified to take advantage of these features.

11.2 Current state-of-the-art

Due to its high bandwidth, low latency, open specification, multi-vendor support, and a roadmap for future speed increases, Infiniband Architecture is perceived as a potential unifying fabric for a high performance computing and storage system interconnection. Currently the Infiniband software has been implemented and optimized with a focus on commercial storage area networks and data centers. This work has led to several limited capability subnet managers that are available from the commercial and open source suppliers. These subnet managers are suitable to small fabrics, but are not usable or scalable for thousands of nodes. The subnet manager and kernel drivers need to be redesigned and modified with a specific focus on ASCI scale machines. The current Infiniband network driver stack consists of over ten kernel modules, each requiring a significant memory footprint. The portability of these drivers is typically limited to a single Linux kernel version, which may be incompatible with other system requirements. For example, parallel cluster file system client drivers typically place specific requirements on the Linux kernel. Current MPI implementations utilize IB RDMA to obtain high performance. However, there is still scope to improve performance by further tuning and utilization of additional IB features, such as multicast, atomic operations, and service levels and types.

11.3 Areas for greatest impact

The areas of potential impact of OSS on Infiniband are listed from highest to lowest impact. The impact is estimated as the risk and usefulness (impact = risk * usefulness) of each area, where risk is defined as the chance the software is not developed without PathForward intervention. Scalable is defined as up to 32,000 nodes.

- Scalable and fast Infiniband subnet manager
- Infiniband drivers improved to achieve lower latency
- Scalable Subnet Manager and Fabric Simulator
- MPI-2 utilizing IBA features for Collective Communications
- Fabric Management optimized for HPC Cluster and Storage systems
- Infiniband routing algorithms optimized for HPC

The potential impact reflects the prioritization each area will receive when funding is allocated. We expect responses to this RFI to address some or all of these areas.

11.4 Area specific requests for information, if needed

We seek responses to the questions in section 8.4 (Open Source System Test/Certification Facility) as appropriate. For more information see the DOE Infiniband Workshop summary document at http://hpcn.ca.sandia.gov/ibworkshop/summary.pdf.

Indicate how the development activity will track changes to Linux and subsystems associated with a common Linux distribution (e.g., United Linux or RedHat) Indicate how the development activity will integrate with compilers developed in Appendix D and application performance analysis and correctness tools developed in Appendix E.

12 Appendix G – Visualization tools and toolkits

Responses are sought for the development of tools and systems components that support scalable visualization on distributed memory systems with COTS graphics. The datasets generated by ASCI simulations put unique scaling demands on visualization systems, both software and hardware. The proposed software systems should aid in one or more facets of the end-to-end visualization process. Responses for scalable system libraries/APIs, visualization toolkits, applications and their comprehensive integration and packaging are encouraged. Responses that include system software features that enable interactive use and dynamic management of resources within a visualization system or end-to-end visualization environment are also desired.

12.1 Background/Requirements

ASCI simulations are producing some of the largest datasets ever computed, and their size continues to increase. Visualization tools and systems have not met this scalability challenge. The ASCI VIEWS (Visual Interactive Environment for Weapons Simulation) program has been deploying a series of prototype systems based on Open Source software and COTS hardware technologies, looking to scale such infrastructures to meet

the challenges of visual data analysis posed by multi-terabyte datasets. These datasets take the form of time varying multi-million cell unstructured meshes with scalar, vector and tensor fields. These datasets and the distributed COTS cluster infrastructures present unique problems for data access/management, visualization algorithms and rendering. Visualization clusters also have the additional task of video delivery over a distance. The remote display may be a mobile device, a desktop, a stereo display or a high resolution "PowerWall".

Visualization imposes certain unique requirements for resource management. Visualization applications are, most often, inherently interactive due to the exploratory and iterative nature of the analysis process. System resources must be dynamically partitionable, schedulable and allocatable for interactive use. The system as a whole must be sharable by simultaneous users who can access resources on demand, constrained by availability. The end-to-end process for computation and data analysis also suggests the need for scheduling and use of resources in combination. For example, if data analysis is to be performed while a computational simulation is in progress, it should be possible to schedule computational and visualization resources for simultaneous use. In this vein, it is also desirable to be able to connect/disconnect visualization applications to ongoing computations, as well as the ability to instantiate and manage distributed cooperating parallel applications (e.g., a computation application and a visualization application). These suggest the availability of dynamic process management and connectivity features.

12.2 Current state-of-the-art

There are a number of Open Source efforts that target various aspects of these problems. In particular, the Chromium project (chromium.sourceforge.net) addresses aspects of the scalable rendering problem while the DMX project (dmx.sourceforge.net) provides distributed X11 services for PowerWalls. The integration of these systems yields an end-to-end parallel rendering infrastructure, targeting COTS clusters.

The VTK toolkit (/public.kitware.com/VTK/) has been used as the underlying data manipulation and visualization library for several applications. Open Source applications like ParaView and VisIt, based on VTK, are providing some of the basic end-user visualization interfaces that are scaling to the desired levels. Other tools such as OpenDX provide similar feature sets and support distributed parallel visualization.

12.3 Areas for greatest impact

COTS clusters are making use of very high performance interconnects (e.g. GigE), from the cluster nodes to the desktop. Responses for systems capable of exploiting these digital paths for remote image delivery would have a large impact. In particular, systems that are capable of exploiting asynchronous image capture (e.g. over DVI) would be valuable.

Scalable distributed visualization frameworks and systems are a major area of interest. Open Source solutions that provide visualization services (data access, data manipulation, primitive generation and visualization) that can be shown to scale to 1000's of nodes would have a large impact, particularly if these frameworks or systems can be easily integrated into systems like VTK. Some specific areas include load balancing and intelligent data layouts and caching.

The clusters being deployed rely on commodity graphics cards for their image generation capabilities. These cards have features that could be used to accelerate specific rendering paths, via features like fragment programming. Systems that are capable of exploiting this largely untapped resource in the context of a distributed visualization infrastructure are of interest. One example would be vertex programs that directly support visualization needs, such as expanding from sphere position/radius into fully tessellated models, thereby alleviating bandwidth requirements onto the card, but leveraging the increasingly powerful Graphics Processing Units (GPUs).

Visualization cluster software configuration and packaging is another area of interest. Many of the existing systems can be difficult to install and optimally configure into a given cluster environment. Software that provides higher-level cluster configuration management, specifically providing management of visualization, rendering and display resources would be very valuable. For example, systems to control video switching, routing and display configuration and systems to configure and control PowerWalls in the context of graphics subsystems like DMX and Chromium are desirable.

12.4 Area specific requests for information, if needed

Indicate how the development activity will track changes to Linux and subsystems associated with a common Linux distribution (e.g., United Linux or RedHat). Indicate how the development activity will integrate with compilers developed in Appendix D and application performance analysis and correctness tools developed in Appendix E. Indicate how the development activity will integrate with the resource managers developed in Appendix B.